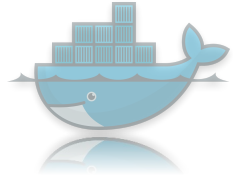
Fundamentos de Docker

Codescrum

Santiago Hyun Dorado

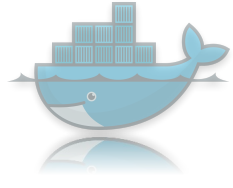
Agenda

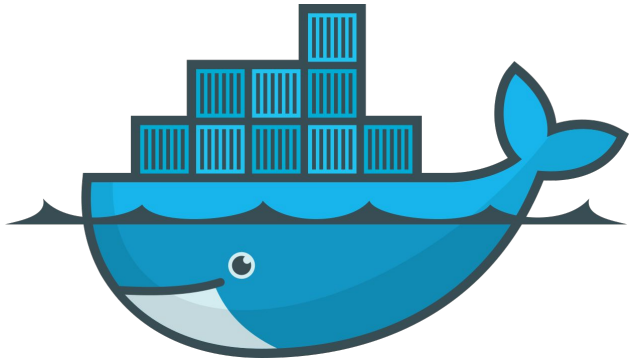
1. Introducción
   - ¿Qué es Docker?
   - ¿Que son contenedores e imágenes?
   - Ventajas de contenedores
2. Arquitectura de Docker
   - Diferencia entre una máquina virtual y un contenedor
   - Componentes
   - Interacción entre componentes
3. Workshop
   - Instalación
   - Ejemplos



Propósito del taller

Comprender y desarrollar conceptos básicos sobre despliegue y ejecución de ambientes en contenedores a través de una de las plataformas tecnológicas más utilizadas en la actualidad.

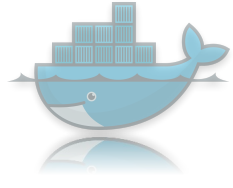




Introducción a Docker

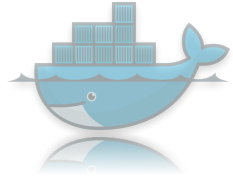
¿Que es Docker?

Docker es una plataforma para desarrolladores y administradores de sistemas que permite desarrollar, desplegar y ejecutar aplicaciones con contenedores.



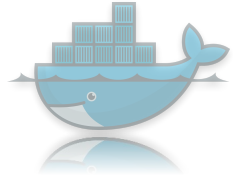
¿Que es un Contenedor?

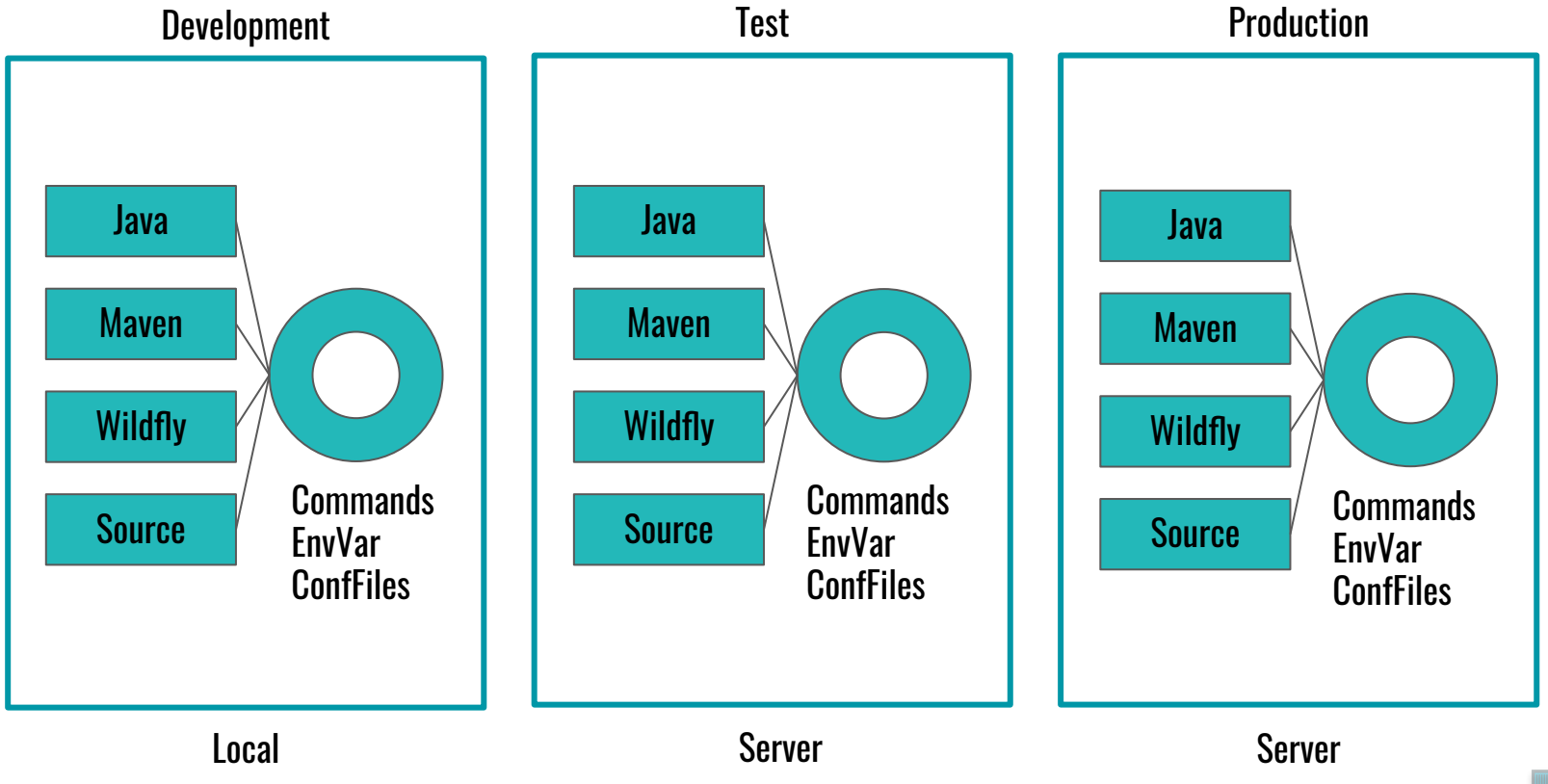
Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable de un entorno informático a otro.

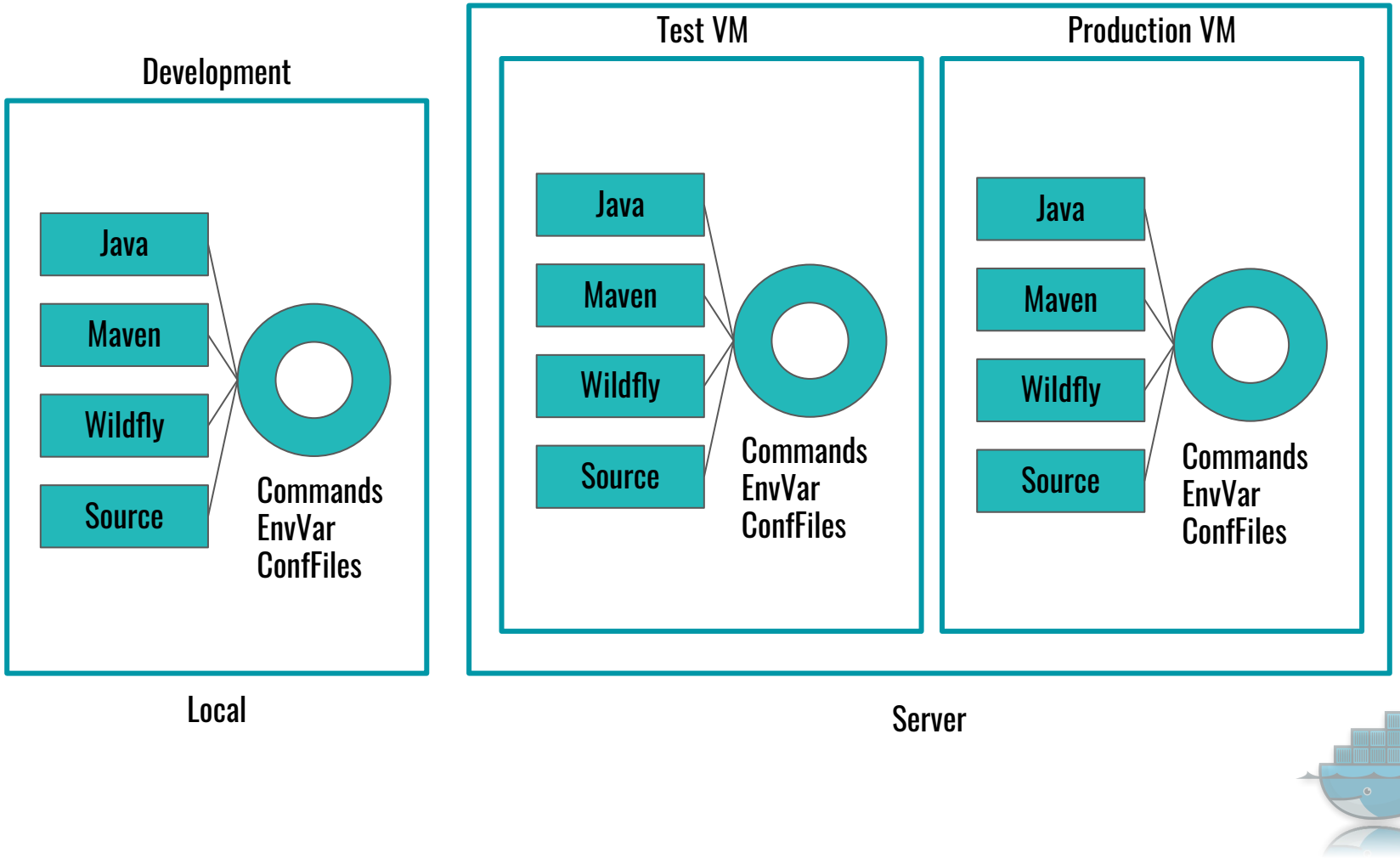


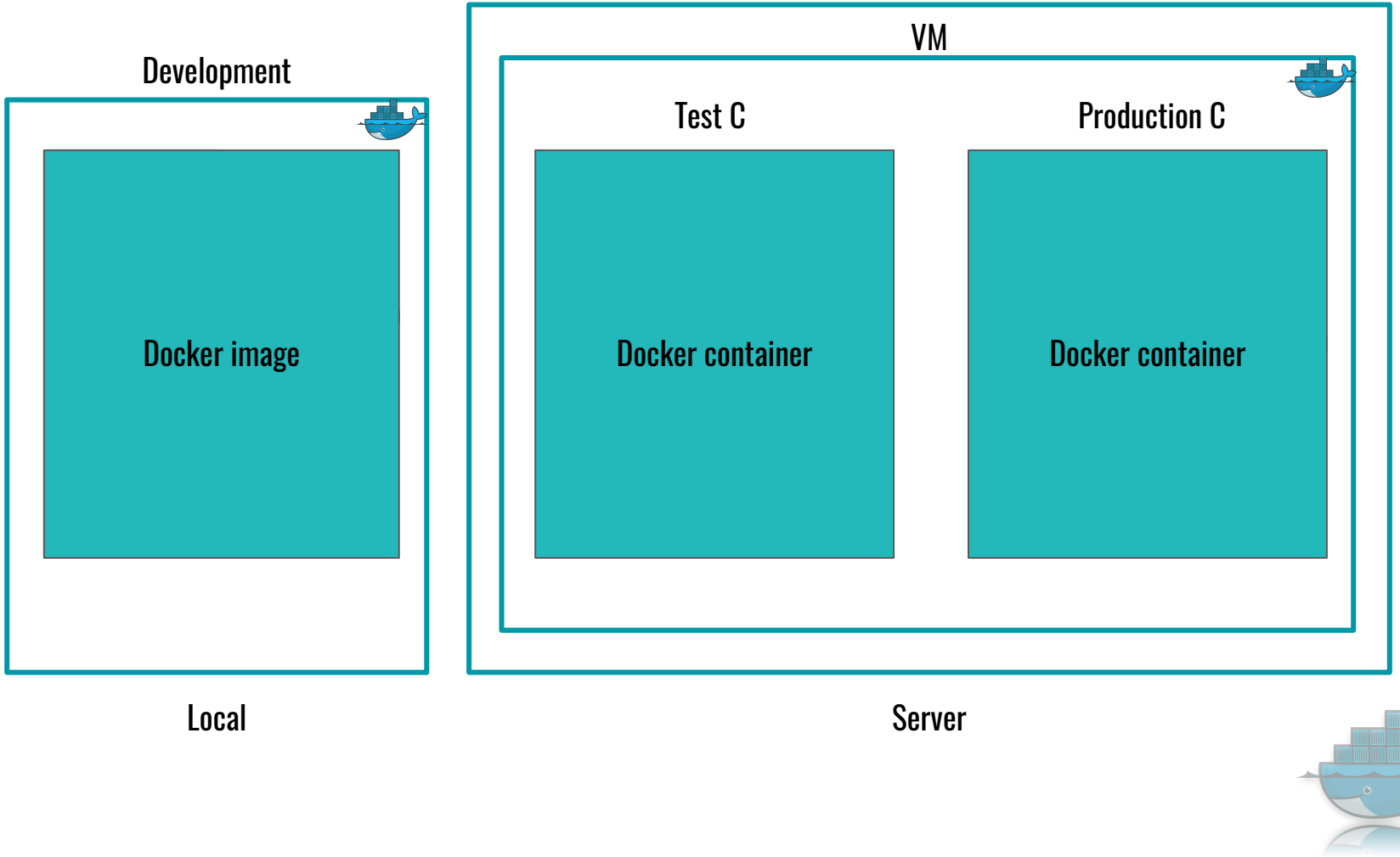
¿Que es una Imagen?

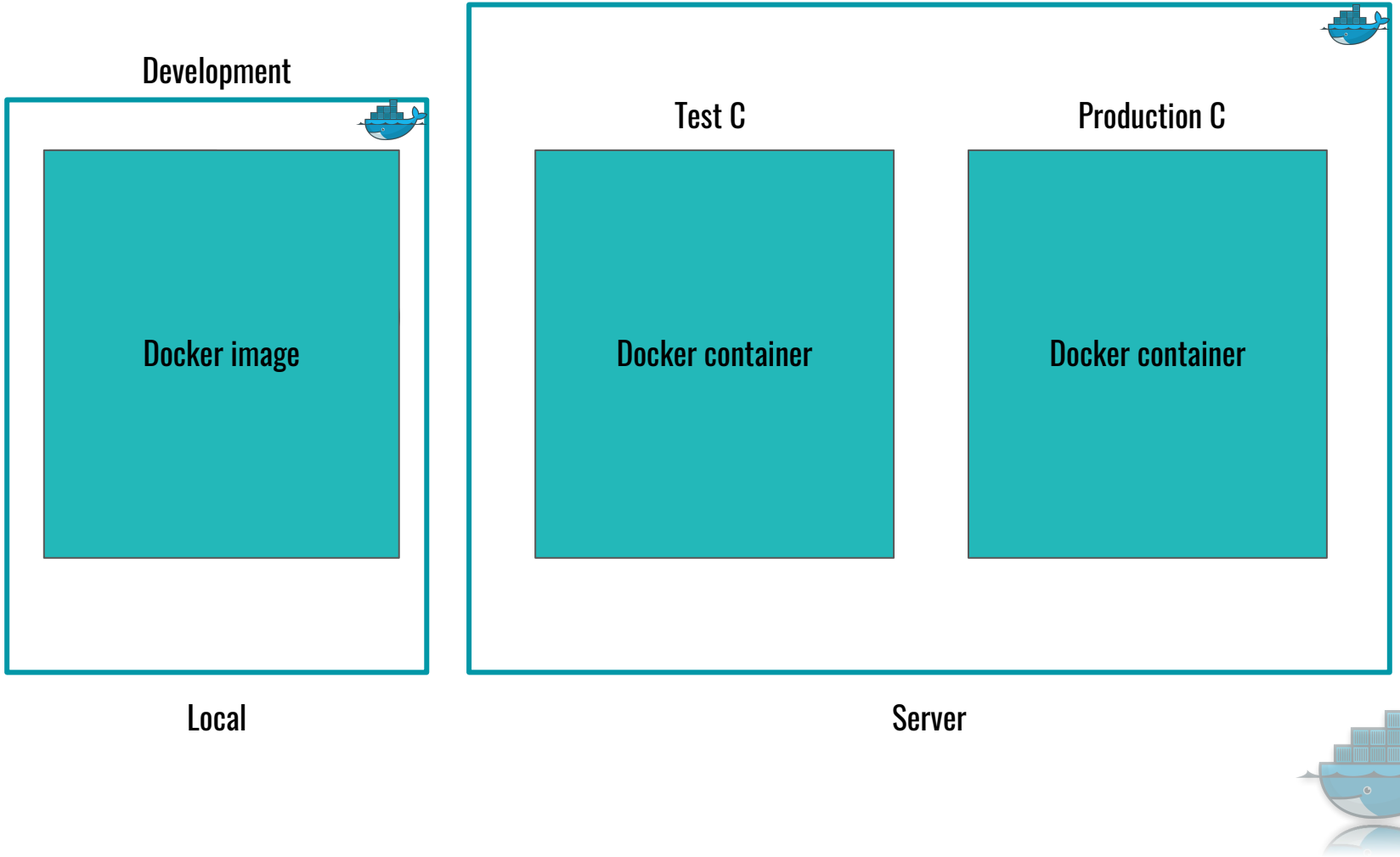
Una imagen es un paquete ejecutable que incluye todo lo necesario para ejecutar una aplicación: el código, el tiempo de ejecución, las librerías, las variables de entorno y los archivos de configuración.




Development
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Local
Test
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Server
Production
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Server

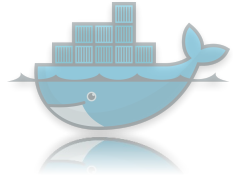

Development
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Local
Test VM
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Production VM
Java
Maven
Wildfly
Source
Commands
EnvVar
ConfFiles
Server

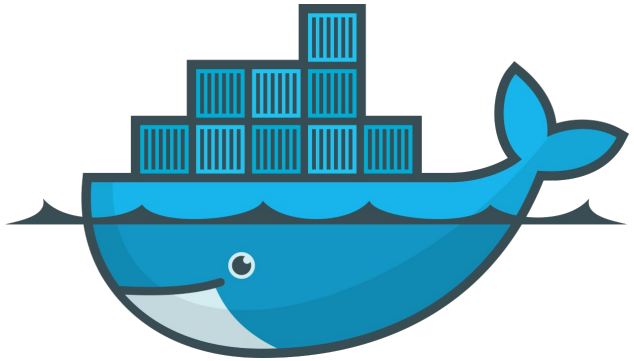

Development
Docker image
Local
VM
Test C
Docker container
Production C
Docker container
Server


Development
Docker image
Local
Test C
Production C
Docker container
Docker container
Server

Ventajas de los contenedores

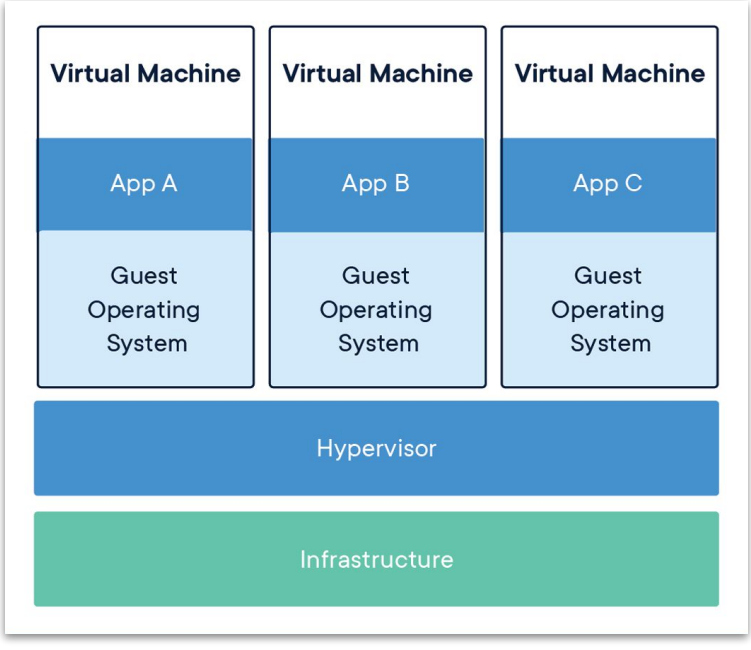
- **Flexible**: Incluso las aplicaciones más complejas se pueden transportar en contenedores.
- **Ligero**: los contenedores aprovechan y comparten el núcleo del host.
- **Intercambiable**: puede implementar actualizaciones y cambios de versión sobre la marcha.
- **Portable**: puede compilar localmente, implementarlo en la nube y ejecutar en cualquier lugar.
- **Escalable**: puede aumentar y distribuir automáticamente réplicas de contenedor.
- **Apilable**: puede apilar servicios verticalmente y sobre la marcha.

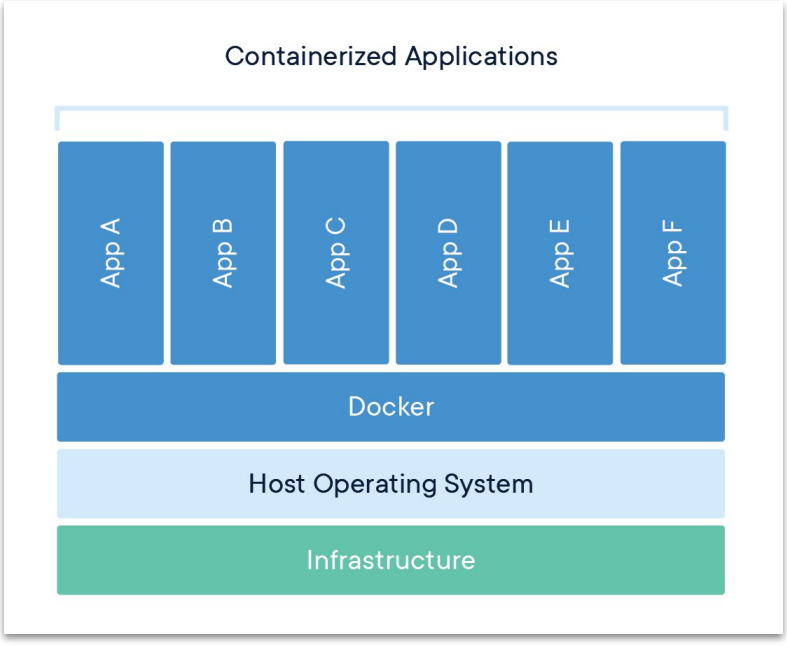




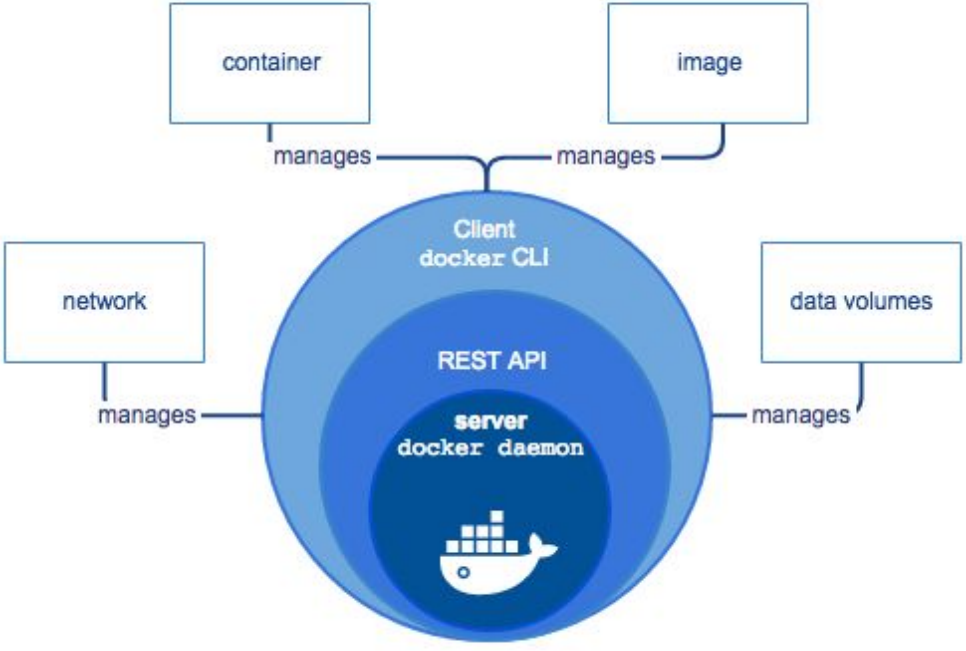
Arquitectura de Docker

Diferencia entre una máquina virtual y un contenedor

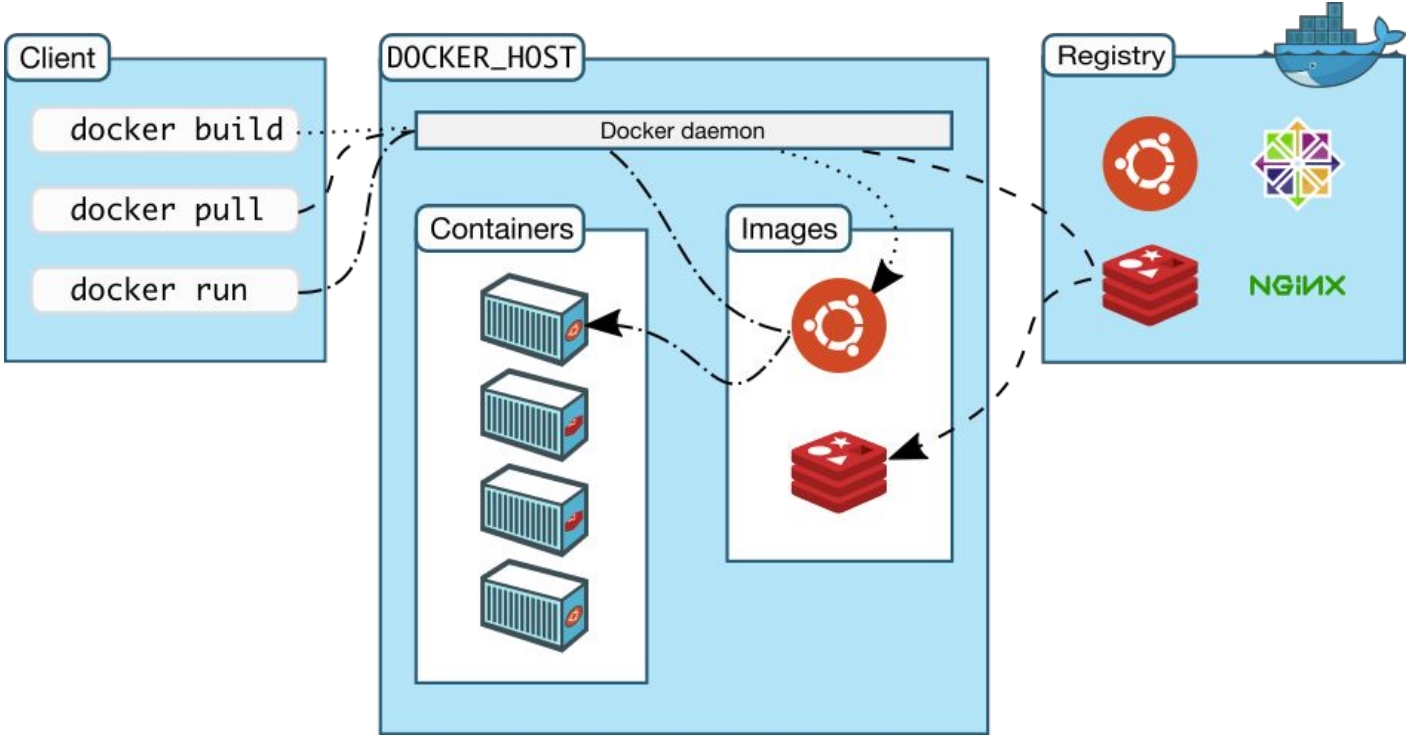


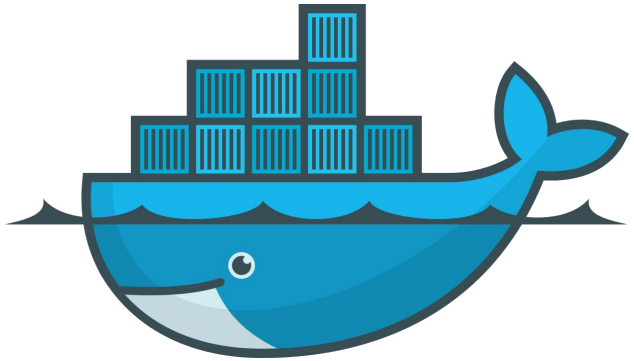


Componentes de Docker



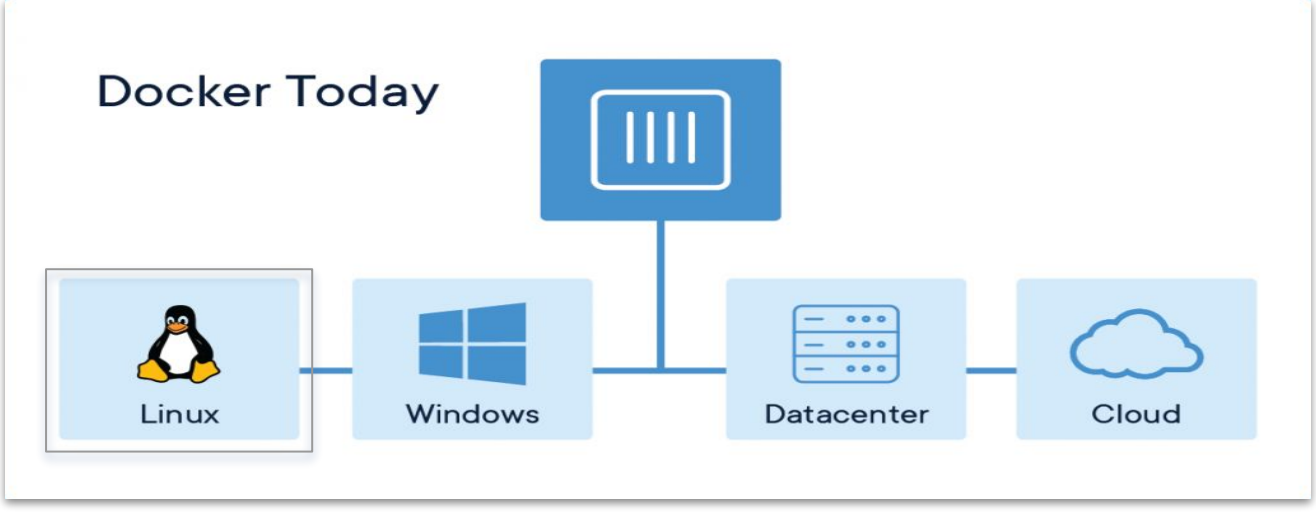
Interacción entre componentes





Instalación y ejemplos

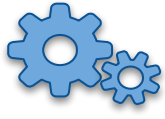
Instalación



Ejemplos - Comandos básicos para Imágenes

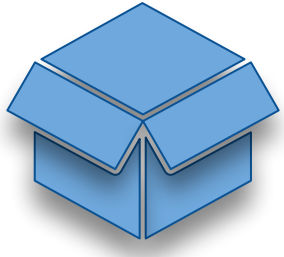
Construcción

docker image build



Carga

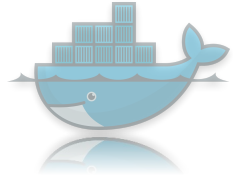
docker image push



Docker Hub (Global registry)

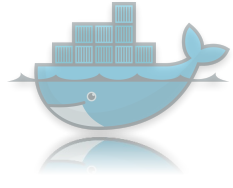
Descarga

docker image pull



Ejemplos - Comandos básicos para Contenedores

1. **Listar**
   docker container ls -a
2. **Construir**
   docker container run <docker-image>
3. **Iniciar**
   docker container start <docker-container>
4. **Detener**
   docker container stop <docker-container>
5. **Eliminar**
   Docker container rm <docker-container>

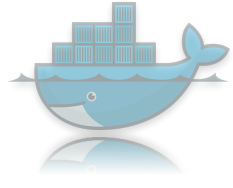


Ejemplo 1

Ejecutando un sitio web estático en un contenedor:

Requisitos:

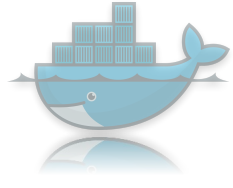
- Código fuente
- Git
- Docker CE
- Cuenta de DockerHub



Ejemplo 1 - Construcción de la imagen

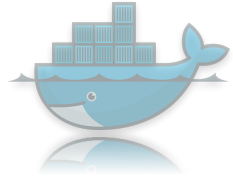
1. **Descargar código fuente**
   git clone https://github.com/zahydo/docker-example-1
2. **Ingresar al directorio**
   cd docker-example-1
3. **Crear Dockerfile**
   touch Dockerfile
4. **Escribir comandos**
   FROM nginx
   COPY . /usr/share/nginx/html
5. **Construir imagen**
   docker image build -t docker-example-1:1.0.0 .

Un Dockerfile es un documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen.



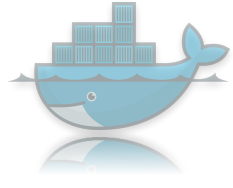
Ejemplo 1 - Carga y descarga del Registro

6. **Crear repositorio en el registro**
   Ingresar a Docker hub y crear el repositorio: docker-example-1
7. **Etiquetar imagen**
   docker image tag docker-example-1:1.0.0 <DockerID>/docker-example-1:1.0.0
8. **Cargar imagen al registro**
   docker image push <DockerID>/docker-example-1:1.0.0
9. **Descargar imagen del registro**
   docker image pull <DockerID>/docker-example-1:1.0.0
10. **Listar imágenes locales**
    docker image ls || docker images



Ejemplo 1 - Ejecución y acceso del contenedor

11. **Ejecutar contenedor**
    docker container run --help
    docker container run -d -p 8081:80 --name example-1 <DockerID>/docker-example-1:1.0.0
12. **Acceder a contenedor en ejecución**
    docker container exec --help
    docker container exec -it example-1 bash
13. **Detener contenedor**
    docker container stop example-1
14. **Iniciar contenedor**
    docker container start example-1
15. **Eliminar contenedor**
    docker container rm example-1

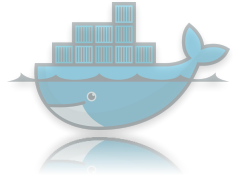


Ejemplo 2

Ejecutando una aplicación en Ruby on Rails:

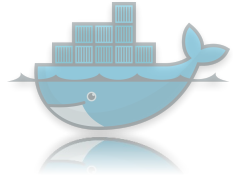
Requisitos:

- Código fuente
- Git
- Docker CE
- Cuenta de DockerHub



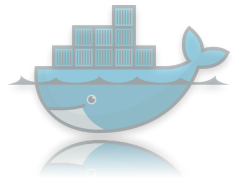
Ejemplo 2 - Despliegue en ‘development’

1. **Descargar código fuente**
   git clone https://github.com/zahydo/docker-example-2.git
2. **Ingresar al directorio**
   cd docker-example-2
3. **Construir imagen**
   docker image build -t docker-example-2 .
4. **Correr contenedor**
   docker container run --name example-2-dev -p 3000:3000 -d docker-example-2
5. **Configurar BD en contenedor**
   docker container exec -it example-2-dev bash
   rake db:setup
   rake db:migrate
   rake db:seed
6. **Observar log**
   docker container logs -f example-2-dev



Ejemplo 2 - Despliegue en ‘production’

1. **Ejecutar contenedor de Postgres**
   - *docker container run --name postgres-db -p 5432:5432 -e POSTGRES_USER=developer -e POSTGRES_PASSWORD=developer_pass -d postgres*
2. **Consultar Ip del contenedor de BD**
   docker container inspect postgres-db
3. **Ejecutar contenedor de la App**
   - *docker run --name example-2-prod -p 3001:3000 -e DATABASE_HOST=<IpAddress> -e DATABASE_USERNAME=developer -e DATABASE_PASSWORD=developer_pass -e RAILS_ENV=production -e SECRET_KEY_BASE=e068f5ecf5504b39cdade0069c02a514794e6c2b9878e74aa8aebce1b839e7201376f168a265eef14d058ccffe1ac0dc8ebf46def69888e5246225e397b71023 -d docker-example-2*
   - *docker run --name example-2-prod -p 3001:3000 --env-file config.env -d docker-example-2*



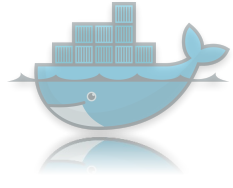
Ejemplo 2 - Despliegue en ‘production’

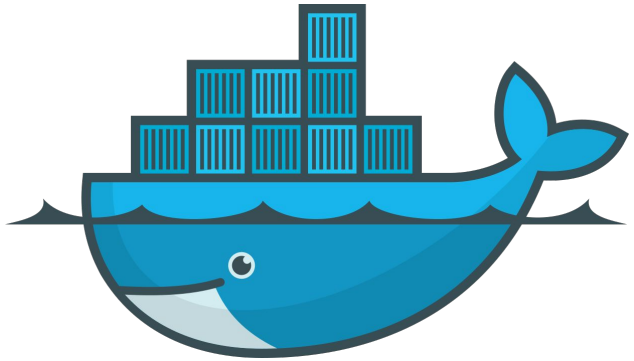
5. **Configurar BD en contenedor**

   docker container exec -it example-2-prod bash
   rake db:setup
   rake db:migrate
   rake db:seed

6. **Observar log**

   docker container exec -it example-2-prod tail -f log/production.log





¡GRACIAS!